Lecture 1: Fundamental Statistical Learning Theory

Xin Xing
VT

Introduction to Statistical Learning Theory

- Machine Learning deals with systems trained from data rather than explicit programming
- Core concept: Learning from examples paradigm
- Primary goal: Find underlying input-output relation

$$f(x_{\text{new}}) \sim y$$

- Two main tasks in statistical learning:
  - Prediction: Estimate outputs for new inputs
  - Inference: Understand relationships in data

Real-World Learning Problems

Examples from Practice

- **Medical:** Predict second heart attack risk from demographic, diet, and clinical measurements
- **Finance:** Predict stock prices based on company performance and economic data
- **Recognition:** Identify handwritten ZIP code digits from 16×16 pixel images
- **Biology:** Estimate blood glucose from infrared absorption spectrum
- **Epidemiology:** Identify risk factors for prostate cancer

Supervised vs. Unsupervised Learning

Supervised Learning

- Outcome variable guides learning
- Have (x_i, y_i) pairs
- Goal: Predict y from x
- Examples:
  - Regression
  - Classification

Unsupervised Learning

- No outcome variable
- Only observe features x_i
- Goal: Discover structure
- Examples:
  - Clustering
  - Dimensionality reduction

Example 1: Email Spam Classification

- Data: 4601 email messages with 57 word/character frequencies
- Task: Classify as spam or legitimate email
- This is a **classification** problem

Sample Decision Rules

- Simple rule: `if (%george < 0.6) & (%you > 1.5) then spam`
- Linear rule: `if (0.2·%you - 0.3·%george) > 0 then spam`

Note

Not all errors are equal: filtering good email is worse than letting spam through

Example 2: Prostate Cancer Regression

- Goal: Predict log PSA from clinical measurements
- Predictors:
  - `lcavol`: log cancer volume
  - `lweight`: log prostate weight
  - `age`: patient age
  - `lbph`: log benign prostatic hyperplasia
  - `svi`: seminal vesicle invasion
  - `lcp`: log capsular penetration
  - `gleason`: Gleason score
- This is a **regression** problem (quantitative outcome)

Statistical Learning Task 1: Prediction

Goal

Given new input x_{new}, predict output y_{new} accurately

- Characteristics:
  - Focus on accuracy of predictions
  - Model interpretability is secondary
  - Often uses complex models (e.g., deep learning)
- Examples:
  - Spam email detection
  - Image recognition
  - Weather forecasting
- Key metric: Prediction error on new data

$$\text{Error} = \mathbb{E}[(y_{\text{new}} - \hat{f}(x_{\text{new}}))^2]$$

Statistical Learning Task 2: Inference

Goal

Understand how output Y is affected by input X

- Characteristics:
  - Focus on relationships between variables
  - Model interpretability is crucial
  - Prefer simpler, interpretable models
- Key Questions:
  - Which variables are important?
  - What is the relationship between variables?
  - How strong is the evidence for these relationships?
- Examples:
  - Effect of diet on health outcomes
  - Impact of education on income
  - Identifying risk factors in disease

Comparison: Prediction vs. Inference

Prediction

- Black box models acceptable
- Focus on accuracy
- Less concerned with "why"
- Examples:
  - Neural networks
  - Random forests

Inference

- Transparent models required
- Focus on understanding
- "Why" is crucial
- Examples:
  - Linear regression
  - Logistic regression

Trade-off: Predictive accuracy vs. Interpretability

Variable Types and Terminology

Inputs (Predictors/Features)

- Denoted by X
- Can be quantitative or qualitative
- $X \in \mathbb{R}^p$ for p predictors

Outputs (Responses)

- Quantitative $Y \in \mathbb{R} \rightarrow$ **Regression**
- Qualitative $G \in \{1, \ldots, K\} \rightarrow$ **Classification**

Coding Qualitative Variables

- Binary: $Y \in \{0, 1\}$ or $\{-1, 1\}$
- Multi-class: Dummy variables (one-hot encoding)

Training Data Structure

- Training set definition:

$$S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

- Where:
  - (x_i, y_i) are input-output pairs
  - Each pair represents an example/sample/data point
- Model requirements:
  - Must account for task uncertainty
  - Must account for data uncertainty

Probabilistic Data Model

- Space Definitions:
  - Input space: $X \subseteq \mathbb{R}^D$
  - Output space Y depends on task:
    - Regression: $Y \subseteq \mathbb{R}$
    - Binary classification: $Y = \{-1, 1\}$
    - Multiclass: $Y = \{1, 2, \ldots, T\}$
- Data Distribution:

$$p(x, y) = p_X(x)p(y|x)$$

  - $p(y|x)$: conditional distribution
  - $p_X(x)$: marginal distribution

Examples of Data Models

Example 1: Regression

$$y = f^*(x) + \epsilon$$

where:

- f^* is unknown function (e.g., $f^*(x) = x^T w^*$)
- $\epsilon \sim \mathcal{N}(0, \sigma)$ is Gaussian noise

Example 2: Binary Classification

Mixture of two Gaussians:

$$p(x|y = -1) = \frac{1}{Z}\mathcal{N}(-1, \sigma_-)$$

$$p(x|y = 1) = \frac{1}{Z}\mathcal{N}(+1, \sigma_+)$$

Noiseless case: $p(1|x) = 1$ or 0

Loss Function and Expected Risk

- Loss Function:

$$\ell : Y \times Y \to [0, \infty)$$

  Measures error $\ell(y, f(x))$ when predicting $f(x)$ instead of y

- Expected Risk (Expected Prediction Error):

$$E(f) = E[\ell(y, f(x))] = \int dxdy\, p(x, y)\ell(y, f(x))$$

- Target Function:
  - $f^* : X \to Y$ minimizes expected risk
  - Cannot be directly computed (p unknown)
  - Goal: Find a solution that generalizes well

Statistical Decision Theory

- For squared error loss $L(Y, f(X)) = (Y - f(X))^2$:

$$\text{EPE}(f) = E_X E_{Y|X} \left[(Y - f(X))^2 | X \right]$$

- Minimizing pointwise:

$$f(x) = \arg\min_c E_{Y|X} \left[(Y - c)^2 | X = x \right]$$

- **Solution:** The conditional expectation

$$f(x) = E(Y|X = x)$$

This is the **regression function**

Decision Theory for Classification

- For 0-1 loss (misclassification):

$$\text{EPE} = E_X \sum_{k=1}^{K} L(G_k, \hat{G}(X)) \Pr(G_k|X)$$

- **Bayes Classifier:**

$$\hat{G}(x) = \arg\max_{g \in \mathcal{G}} \Pr(g|X = x)$$

  Classify to the most probable class

- **Bayes Error Rate:** The irreducible error rate of the Bayes classifier

Two Simple Approaches to Prediction

Contrasting Methods

1. **Linear Model (Least Squares)**
   - Strong structural assumptions
   - Stable but potentially biased
2. **K-Nearest Neighbors**
   - Mild structural assumptions
   - Flexible but potentially unstable

These represent opposite ends of a spectrum

Linear Models and Least Squares

- Linear model:

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j = X^T \hat{\beta}$$

- Fit by minimizing Residual Sum of Squares:

$$\text{RSS}(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

- **Solution:**

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- Decision boundary: $\{x : x^T \hat{\beta} = 0.5\}$ (linear/hyperplane)

Nearest-Neighbor Methods

- K-nearest neighbor estimate:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  where $N_k(x)$ is the neighborhood of k closest points to x
- Two approximations:
  1. Expectation $\approx$ sample average
  2. Conditioning at a point $\approx$ conditioning on a neighborhood
- As $N, k \to \infty$ with $k/N \to 0$:

$$\hat{f}(x) \to E(Y|X = x)$$

Linear vs. KNN: Key Differences

Least Squares

- Assumes $f(x) \approx x^T \beta$ globally
- Low variance
- Potentially high bias
- Decision boundary: smooth, linear
- Few parameters (p)

K-Nearest Neighbors

- Assumes f locally constant
- High variance (small k)
- Low bias (small k)
- Decision boundary: irregular
- Effective parameters: N/k

Bias-Variance Decomposition: Introduction

Key Question

How to choose optimal K in K-Nearest Neighbors?

- Goal: Minimize expected error

$$\mathbb{E}_S \mathbb{E}_{x,y}(y - \hat{f}_K(x))^2$$

- Consider regression model:

$$y_i = f^*(x_i) + \delta_i$$

where:

- δ_i is independent with x_i
- $\mathbb{E}[\delta_i] = 0$
- $\mathbb{E}[\delta_i^2] = \sigma^2$

Two Sources of Randomness

Why Two Layers of Expectation?

$$\text{EPE} = \mathbb{E}_{\mathcal{S}}\mathbb{E}_{x,y}(y - \hat{f}(x))^2$$

There are **two distinct sources of randomness**:

1. **Training set** $\mathcal{S} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$
   - Different training sets $\rightarrow$ different fitted models $\hat{f}$
   - $\mathbb{E}_{\mathcal{S}}$ averages over all possible training sets
2. **Test point** (x, y)
   - The new observation we want to predict
   - $\mathbb{E}_{x,y}$ averages over all possible test points

Visualizing the Two Sources

Thought Experiment:

1. Draw a training set $\mathcal{S}$ from population
2. Fit model: $\mathcal{S} \rightarrow \hat{f}_{\mathcal{S}}$
3. Draw a test point (x, y) from same population
4. Compute squared error: $(y - \hat{f}_{\mathcal{S}}(x))^2$
5. **Repeat many times** and average

The Two Expectations Correspond To:

- **Inner** $\mathbb{E}_{x,y}$: Average over all test points (for fixed training set)
- **Outer** $\mathbb{E}_{\mathcal{S}}$: Average over all possible training sets

$$\underbrace{\mathbb{E}_{\mathcal{S}}}_{\text{training sets}} \quad \underbrace{\mathbb{E}_{x,y}}_{\text{test points}} \quad (y - \hat{f}_{\mathcal{S}}(x))^2$$

Why Separate the Expectations?

The Predictor $\hat{f}$ is a Random Variable!

$\hat{f}$ depends on the training set $\mathcal{S}$, which is random.

- **Fixed $\mathcal{S}$:** Given a specific training set, $\hat{f}$ is fixed
  - Can compute $\mathbb{E}_{x,y}[(y - \hat{f}(x))^2 | \mathcal{S}]$
  - This measures error for *this particular* fitted model
- **Random $\mathcal{S}$:** The model $\hat{f}$ varies across training sets
  - $\mathbb{E}_{\mathcal{S}}[\cdot]$ captures this variability
  - This is the **variance** of the learning algorithm

Key Insight

Separating the expectations allows us to decompose error into **bias** (systematic) and **variance** (due to training set randomness).

Decomposing $\mathbb{E}_{x,y}$ Further

- The test point (x, y) also has two components:
  - x: the input features
  - $y|x$: the output given the input
- We can write:

$$\mathbb{E}_{x,y} = \mathbb{E}_x \mathbb{E}_{y|x}$$

- Full decomposition:

$$\text{EPE} = \mathbb{E}_{\mathcal{S}} \mathbb{E}_x \mathbb{E}_{y|x} (y - \hat{f}(x))^2$$

Three Sources of Randomness

1. $\mathcal{S}$: Which training data we observe
2. x: Where we want to predict
3. $y|x$: Noise in the response $(y = f^*(x) + \varepsilon)$

The Order of Expectations Matters

- **Pointwise analysis:** Fix x, analyze error at that point

$$\text{EPE}(x) = \mathbb{E}_{\mathcal{S}}\mathbb{E}_{y|x}(y - \hat{f}(x))^2$$

- **Then average over x:**

$$\text{EPE} = \mathbb{E}_x[\text{EPE}(x)]$$

Pointwise Decomposition (at fixed x)

$$\text{EPE}(x) = \underbrace{\sigma^2}_{\text{Irreducible}} + \underbrace{\mathbb{E}_{\mathcal{S}}[(\hat{f}(x) - \mathbb{E}_{\mathcal{S}}[\hat{f}(x)])^2]}_{\text{Variance of } \hat{f}(x)}$$

$$+ \underbrace{(\mathbb{E}_{\mathcal{S}}[\hat{f}(x)] - f^*(x))^2}_{\text{Bias}^2 \text{ of } \hat{f}(x)}$$

This decomposition is possible **because** we separate the expectations!

Summary: Sources of Prediction Error

Component	**Source**	**Can we reduce it?**
Irreducible error	$\text{Var}(y\|x) = \sigma^2$	No (inherent noise)
Bias^2	$\mathbb{E}_{\mathcal{S}}[\hat{f}] \neq f^*$	Yes (more flexible model)
Variance	$\hat{f}$ varies with $\mathcal{S}$	Yes (simpler model, more data)

The Fundamental Trade-off

- More complex models: $\downarrow$ Bias, $\uparrow$ Variance
- Simpler models: $\uparrow$ Bias, $\downarrow$ Variance
- Goal: Find the sweet spot that minimizes **total** error

Error Decomposition

- Expected loss at point x:

$$\varepsilon(K) = \mathbb{E}_x \mathbb{E}_S \mathbb{E}_{y|x}(y - \hat{f}_K(x))^2$$

- Decomposition:

$$\varepsilon(K) = \sigma^2 + \mathbb{E}_S \mathbb{E}_{y|x}(f^*(x) - \hat{f}_K(x))^2$$

- Further breakdown:

$$\mathbb{E}_S \mathbb{E}_{y|x}(f^*(x) - \hat{f}_K(x))^2 = \underbrace{(f^*(x) - \mathbb{E}_S \mathbb{E}_{y|x} \hat{f}_K(x))^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}_S \mathbb{E}_{y|x}(\mathbb{E}_{y|x} \hat{f}_K(x) - \hat{f}_K(x))^2}_{\text{Variance}}$$

Graphical Definition

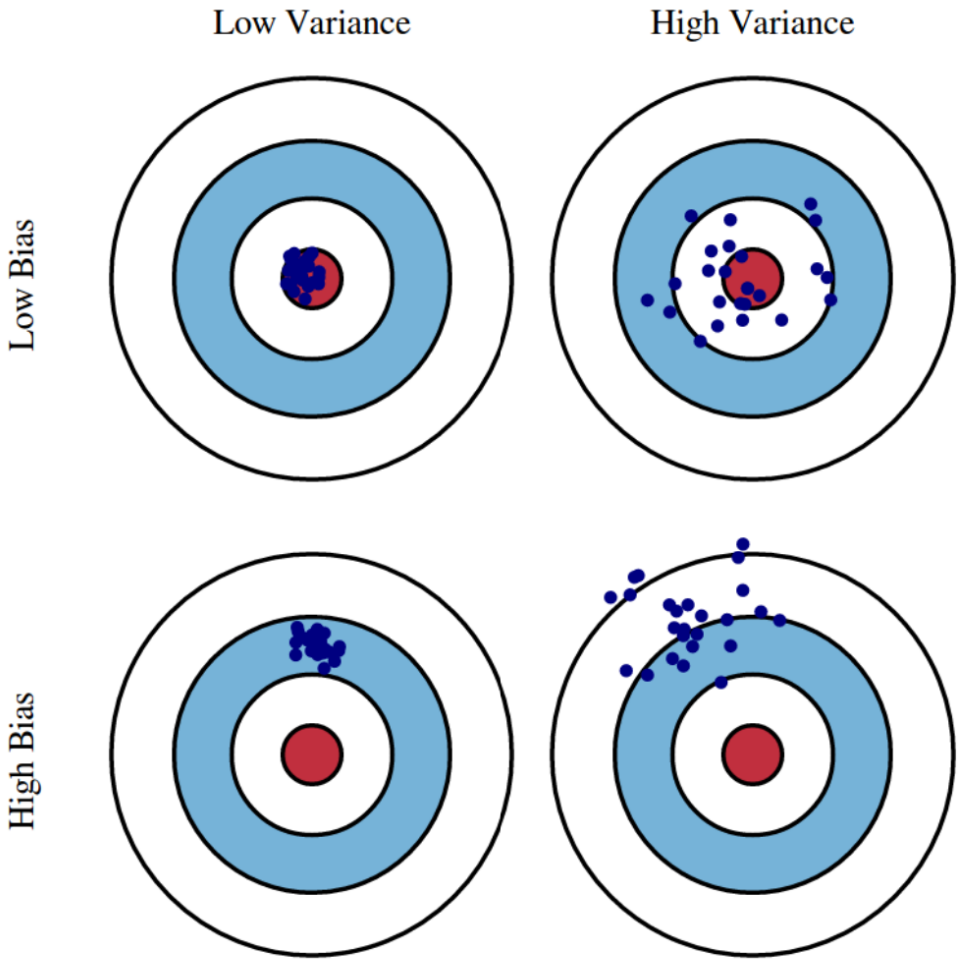

Fig 1: Graphical illustration of bias and variance.
Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

A simple example: KNN Estimator

- Consider KNN estimator at point x:

$$\hat{f}_K(x) = \frac{1}{K} \sum_{\ell \in K_x} y_\ell$$

where:

  - K_x is the set of K nearest neighbors
  - y_ℓ are the corresponding outputs

- Regression model:

$$y_i = f^*(x_i) + \delta_i$$

where:

$$\mathbb{E}[\delta_i] = 0, \quad \mathbb{E}[\delta_i^2] = \sigma^2$$

Expected KNN Algorithm

- Take expectation of $\hat{f}_K(x)$ with respect to $y|x$:

$$\begin{aligned}
\mathbb{E}[\hat{f}_K(x)] &= \mathbb{E}\left[\frac{1}{K}\sum_{\ell \in K_x} y_\ell\right] \\
&= \frac{1}{K}\sum_{\ell \in K_x} \mathbb{E}[y_\ell] \\
&= \frac{1}{K}\sum_{\ell \in K_x} \mathbb{E}[f^*(x_\ell) + \delta_\ell] \\
&= \frac{1}{K}\sum_{\ell \in K_x} f^*(x_\ell)
\end{aligned}$$

where we used $\mathbb{E}[\delta_\ell] = 0$

Bias-Variance Decomposition: Setup

- Consider squared error:

$$\mathbb{E}(f^*(x) - \hat{f}_K(x))^2$$

- Add and subtract expected KNN estimate:

$$\mathbb{E}(f^*(x) - \hat{f}_K(x))^2 =$$
$$\mathbb{E}((f^*(x) - \mathbb{E}\hat{f}_K(x)) + (\mathbb{E}\hat{f}_K(x) - \hat{f}_K(x)))^2$$

Bias-Variance Decomposition: Cross Terms

- Expand squared term:

$$\mathbb{E}(f^*(x) - \hat{f}_K(x))^2 =$$
$$(f^*(x) - \mathbb{E}\hat{f}_K(x))^2 +$$
$$\mathbb{E}(\mathbb{E}\hat{f}_K(x) - \hat{f}_K(x))^2 +$$
$$2\mathbb{E}[(f^*(x) - \mathbb{E}\hat{f}_K(x))(\mathbb{E}\hat{f}_K(x) - \hat{f}_K(x))]$$

- Cross term is zero because:

$$\mathbb{E}[\mathbb{E}\hat{f}_K(x) - \hat{f}_K(x)] = 0$$

Final Decomposition

$$\mathbb{E}(f^*(x) - \hat{f}_K(x))^2 =$$
$$\underbrace{(f^*(x) - \mathbb{E}\hat{f}_K(x))^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}(\mathbb{E}\hat{f}_K(x) - \hat{f}_K(x))^2}_{\text{Variance}}$$

- Bias term: systematic error in estimation
- Variance term: variability due to random sampling

Computing the Variance Term

- Variance calculation:

$$\mathbb{E}(\mathbb{E}\hat{f}_K(x) - \hat{f}_K(x))^2 =$$

$$\mathbb{E}\left(\frac{1}{K}\sum_{\ell \in K_x} \delta_\ell\right)^2 =$$

$$\frac{1}{K^2}\sum_{\ell \in K_x} \mathbb{E}[\delta_\ell^2] =$$

$$\frac{1}{K^2}\sum_{\ell \in K_x} \sigma^2 = \frac{\sigma^2}{K}$$

Final Error Formula

Complete Expression

$$\varepsilon(K) = \sigma^2 + \left(f^*(x) - \frac{1}{K} \sum_{\ell \in K_x} f^*(x_\ell) \right)^2 + \frac{\sigma^2}{K}$$

- Terms:
  - σ^2: irreducible error
  - Middle term: squared bias
  - $\frac{\sigma^2}{K}$: variance
- Trade-off:
  - Increasing K reduces variance
  - But may increase bias if f^* is not constant

Understanding Bias and Variance

Bias

- Systematic error
- May increases with K
- Larger K $\rightarrow$ more averaging
- Further neighbors less representative

Variance

- Random error
- Decreases with K
- Formula: $\frac{\sigma^2}{K}$
- More neighbors $\rightarrow$ more stable

The Bias-Variance Trade-off

- Effect of K on components:
  - Small K:
    - Low bias (close neighbors $\approx$ true value)
    - High variance (sensitive to noise)
  - Large K:
    - Higher bias (averaging distant points)
    - Lower variance (more stable predictions)
- Optimal K balances:
  - Data noise level
  - Function smoothness
  - Training set size

Tradeoff

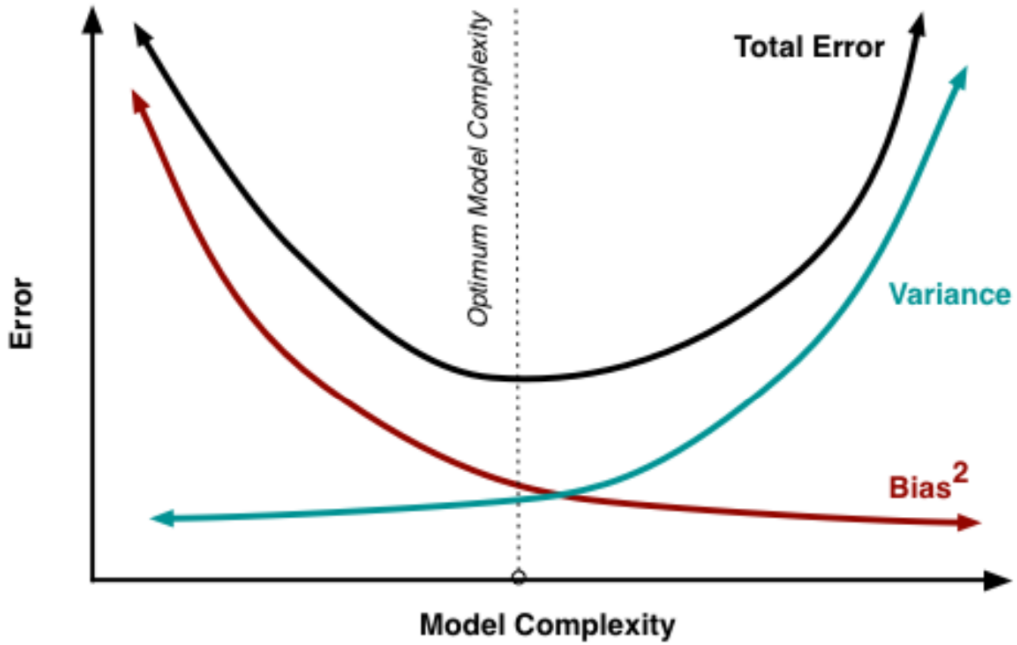

Fig 2: The variation of Bias and Variance with the model complexity. This is similar to the concept of overfitting and underfitting. More complex models overfit while the simplest models underfit.
Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Practical Implications

Key Insights

Choice of K depends on:

- Data characteristics:
  - Noise level (σ^2)
  - Sample size (n)
  - Feature dimensionality (D)
- Trade-offs:
  - Small dataset $\rightarrow$ Larger K (reduce variance)
  - Noisy data $\rightarrow$ Larger K (more averaging)
  - Complex function $\rightarrow$ Smaller K (reduce bias)

The Curse of Dimensionality

Problem

Local methods break down in high dimensions

- To capture fraction r of observations in p-dimensional hypercube:

$$e_p(r) = r^{1/p}$$

  Expected edge length of neighborhood
- Example ($p = 10$):
  - To capture 1% of data: $e_{10}(0.01) = 0.63$ (63% of range!)
  - To capture 10% of data: $e_{10}(0.1) = 0.80$ (80% of range!)
- Neighborhoods are no longer "local"

Median Distance to Closest Point: Setup

Problem Setup

- N data points uniformly distributed in a p-dimensional unit ball
- Center at the origin
- Question: What is the median distance from origin to the **closest** point?

- **Key fact:** For a p-dimensional ball, volume scales as r^p

$$\frac{V_p(r)}{V_p(1)} = r^p$$

- For uniform distribution in unit ball:

$$P(\|X\| \leq r) = r^p \quad \text{for } 0 \leq r \leq 1$$

Median Distance: Derivation (1/2)

- **Step 1:** Probability a single point is farther than r from origin:

$$P(\|X\| > r) = 1 - r^p$$

- **Step 2:** Probability that **all** N points are farther than r:

$$P(\text{all } N \text{ points } > r) = (1 - r^p)^N$$

(points are independent)

- **Step 3:** Probability that the **closest** point is within distance r:

$$P(d_{\min} \leq r) = 1 - (1 - r^p)^N$$

This is the CDF of the minimum distance

Median Distance: Derivation (2/2)

- **Step 4:** Find median d such that $P(d_{\min} \leq d) = \frac{1}{2}$:

$$1 - (1 - d^p)^N = \frac{1}{2}$$

$$(1 - d^p)^N = \frac{1}{2}$$

$$1 - d^p = \left(\frac{1}{2}\right)^{1/N}$$

$$d^p = 1 - \left(\frac{1}{2}\right)^{1/N}$$

$$d = \left(1 - \left(\frac{1}{2}\right)^{1/N}\right)^{1/p}$$

Final Formula

$$d(p, N) = \left(1 - \left(\frac{1}{2}\right)^{1/N}\right)^{1/p}$$

Median Distance: Numerical Examples

Formula

$$d(p, N) = \left(1 - 2^{-1/N}\right)^{1/p}$$

- For $N = 500$ samples:
  - $p = 1$: $d(1, 500) = 1 - 2^{-1/500} \approx 0.00139$
  - $p = 3$: $d(3, 500) \approx 0.11$
  - $p = 10$: $d(10, 500) \approx 0.52$ (halfway to boundary!)
  - $p = 20$: $d(20, 500) \approx 0.72$

Implication

In 10 dimensions, even with 500 training points, the nearest neighbor to the origin is typically more than halfway to the edge of the sample space!

Why This Matters for Learning

- **Near the boundary** $\Rightarrow$ must **extrapolate**, not interpolate
- Extrapolation is much harder and less reliable than interpolation
- The "neighborhood" in KNN becomes nearly the entire space
- **Consequence:** Local averaging fails
  - Bias increases (averaging over non-local region)
  - The assumption "f is locally constant" breaks down

The Curse in One Sentence

In high dimensions, all points are far from all other points, and most points are near the boundary.

Curse of Dimensionality: Implications

- **Sparse sampling:** $N^{1/p}$ is the sampling density
  - Need $N = 100^{10}$ samples in 10D for same density as 100 in 1D
- **Edge effects:** Most points are near the boundary
  - Median distance to closest point:

$$d(p, N) = \left(1 - \frac{1}{2}^{1/N}\right)^{1/p}$$

  - For $N = 500$, $p = 10$: $d \approx 0.52$ (halfway to boundary!)
- **Consequence:** Prediction requires extrapolation, not interpolation

Escaping the Curse of Dimensionality

Key Insight

Impose structure to reduce effective dimensionality

- **Linear models:** Assume $f(x) \approx x^T \beta$
  - EPE grows only linearly: $\sigma^2(p/N) + \sigma^2$
- **Additive models:** $f(X) = \sum_{j=1}^{p} f_j(X_j)$
  - Reduces to p one-dimensional problems
- **General principle:** Trade flexibility for stability through assumptions

Linear Model EPE: Setup

Assume the True Model is Linear

$$Y = X^T\beta + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$$

- Fit by least squares to training data $(\mathbf{X}, \mathbf{y})$
- Least squares estimate:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Prediction at test point x_0:

$$\hat{y}_0 = x_0^T\hat{\beta}$$

- **Goal:** Compute $\text{EPE}(x_0) = E_{y_0|x_0}E_{\mathcal{T}}(y_0 - \hat{y}_0)^2$

Linear Model EPE: Bias-Variance Decomposition

- General decomposition:

$$\begin{aligned}\text{EPE}(x_0) &= E_{y_0|x_0} E_{\mathcal{T}}(y_0 - \hat{y}_0)^2 \\ &= \underbrace{\text{Var}(y_0|x_0)}_{\text{Irreducible}} + \underbrace{E_{\mathcal{T}}[\hat{y}_0 - E_{\mathcal{T}}\hat{y}_0]^2}_{\text{Variance}} + \underbrace{[E_{\mathcal{T}}\hat{y}_0 - x_0^T\beta]^2}_{\text{Bias}^2}\end{aligned}$$

- **Irreducible error:** $\text{Var}(y_0|x_0) = \sigma^2$
- **Key fact:** Least squares is **unbiased** when model is correct!

$$E_{\mathcal{T}}[\hat{\beta}] = \beta \quad \Rightarrow \quad E_{\mathcal{T}}[\hat{y}_0] = x_0^T\beta$$

- Therefore: $\text{Bias}^2 = 0$

Linear Model EPE: Why is $\hat{\beta}$ Unbiased?

- Substitute $\mathbf{y} = \mathbf{X}\beta + \varepsilon$ into $\hat{\beta}$:

$$\begin{aligned}\hat{\beta} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\beta + \varepsilon) \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\beta + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\varepsilon \\ &= \beta + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\varepsilon\end{aligned}$$

- Taking expectation (conditioning on $\mathbf{X}$):

$$E[\hat{\beta}|\mathbf{X}] = \beta + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \underbrace{E[\varepsilon]}_{=0} = \beta$$

Linear Model EPE: Computing the Variance (1/2)

- We need: $\text{Var}_{\mathcal{T}}(\hat{y}_0) = \text{Var}_{\mathcal{T}}(x_0^T \hat{\beta})$
- From previous slide: $\hat{\beta} = \beta + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\varepsilon$
- So: $\hat{y}_0 = x_0^T\hat{\beta} = x_0^T\beta + x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\varepsilon$
- Write as linear combination of errors:

$$\hat{y}_0 = x_0^T\beta + \sum_{i=1}^{N} \ell_i(x_0)\varepsilon_i$$

  where $\ell_i(x_0)$ is the i-th element of $\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}x_0$
- Since ε_i are independent with variance σ^2:

$$\text{Var}(\hat{y}_0|\mathbf{X}) = \sigma^2 \sum_{i=1}^{N} \ell_i(x_0)^2$$

Linear Model EPE: Computing the Variance (2/2)

- In matrix form:
$$\text{Var}(\hat{y}_0|\mathbf{X}) = \sigma^2 x_0^T (\mathbf{X}^T\mathbf{X})^{-1} x_0$$
- **EPE at x_0:**
$$\text{EPE}(x_0) = \sigma^2 + \sigma^2 x_0^T (\mathbf{X}^T\mathbf{X})^{-1} x_0$$
- Now average over test points x_0 and training sets...

Intermediate Result

$$\text{EPE}(x_0) = \underbrace{\sigma^2}_{\text{irreducible}} + \underbrace{\sigma^2 x_0^T (\mathbf{X}^T\mathbf{X})^{-1} x_0}_{\text{variance (depends on } x_0)}$$

Linear Model EPE: Averaging Over Everything

- Assume N is large and $\mathbf{X}$ is random with $E[X] = 0$
- Then: $\frac{1}{N}\mathbf{X}^T\mathbf{X} \to \text{Cov}(X)$
- So: $(\mathbf{X}^T\mathbf{X})^{-1} \approx \frac{1}{N}\text{Cov}(X)^{-1}$
- Average EPE over x_0:

$$E_{x_0}[\text{EPE}(x_0)] \approx \sigma^2 + \frac{\sigma^2}{N} E_{x_0}\left[x_0^T \text{Cov}(X)^{-1} x_0\right]$$

- Using the trace trick: $E[x^T A x] = \text{tr}(A \cdot \text{Cov}(x))$

$$E_{x_0}\left[x_0^T \text{Cov}(X)^{-1} x_0\right] = \text{tr}\left(\text{Cov}(X)^{-1}\text{Cov}(x_0)\right)$$
$$= \text{tr}(I_p) = p$$

Linear Model EPE: Final Result

Expected Prediction Error for Linear Models

$$E_{x_0}[\text{EPE}(x_0)] = \sigma^2 + \frac{p}{N}\sigma^2 = \sigma^2\left(1 + \frac{p}{N}\right)$$

- **Linear growth in p:** EPE increases as $O(p/N)$
- **Compare to KNN:** Suffers from curse of dimensionality (exponential)
- **Implications:**
  - If $N \gg p$: variance term is negligible
  - If σ^2 is small: growth is slow
  - No bias (if model is correct!)

The Price We Pay

This only works **if the linear model is correct**. If the true f is nonlinear, we incur bias that doesn't decrease with N.

Linear vs. KNN: The Complete Picture

	Linear Model	**KNN**
Bias	0 (if correct) High (if wrong)	Low for small k Increases with k
Variance	$\sigma^2 \cdot p/N$ (linear in p)	σ^2/k (but k must grow with p)
Curse?	**No**	**Yes**
Assumption	$f(x) = x^T\beta$	f locally constant

Key Insight

Linear models escape the curse by making **strong global assumptions**, trading potential bias for guaranteed low variance growth.

Model Complexity and Generalization

- **Training error** decreases with complexity
- **Test error** has U-shape:
  - Too simple $\rightarrow$ underfitting (high bias)
  - Too complex $\rightarrow$ overfitting (high variance)

Expected Prediction Error for KNN

$$\text{EPE}_k(x_0) = \sigma^2 + \underbrace{\left[f(x_0) - \frac{1}{k} \sum_{\ell=1}^{k} f(x_{(\ell)}) \right]^2}_{\text{Bias}^2} + \underbrace{\frac{\sigma^2}{k}}_{\text{Variance}}$$

- Model complexity $\propto N/k$ (effective parameters)

Classes of Restricted Estimators

1. **Roughness Penalty / Regularization**

$$\text{PRSS}(f;\lambda) = \text{RSS}(f) + \lambda J(f)$$

Examples: Ridge regression, LASSO, smoothing splines

2. **Kernel Methods and Local Regression**

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)}$$

Examples: KNN, Nadaraya-Watson, local polynomial

3. **Basis Functions / Dictionary Methods**

$$f_\theta(x) = \sum_{m=1}^{M} \theta_m h_m(x)$$

Examples: Polynomials, splines, neural networks

Summary: Key Concepts

- **Supervised learning:** Learn f from (x_i, y_i) pairs
- **Two main tasks:** Prediction vs. Inference
- **Fundamental trade-off:** Bias vs. Variance
- **Model complexity:** Controls the trade-off
- **Curse of dimensionality:** Local methods fail in high-D
- **Solution:** Impose structure (assumptions) on f

The Central Question

How do we choose the right level of model complexity?
$\rightarrow$ Model selection (cross-validation, etc.)